

“Object Oriented Programming”

Course a.a. 2015-2016

Lecturer: Prof. Romina Eramo (romina.eraamo@univaq.it)

Digital Library Deliverable

| | |
|-------------|------------|
| Date | 29/11/2016 |
| Team (Name) | 24CinL |

| Team Members | | |
|---------------------|----------------------|--------------------------------------|
| Name & Surname | Matriculation Number | E-mail address |
| Brian Parmegiani | 227812 | brian.parmegiani@student.univaq.it |
| Davide Romano | 227618 | davide.romano1@student.univaq.it |
| Davide Di Francesco | 229472 | davide.difrancesco@student.univaq.it |

Table of Contents

| | |
|---|-----------|
| 1 Requirements | 3 |
| 1.1 Functional Requirements | 3 |
| 1.2 Assumptions | 4 |
| 1.3 Non Functional Requirements | 4 |
| 2 Use Case | 5 |
| 2.1 Use Case and Actors | 5 |
| 2.2 Use Case Diagram | 6 |
| 2.3 Use Case description | 7 |
| 3 System Design | 9 |
| 3.1 Component Diagram | 9 |
| 3.2 Architectural choices | 10 |
| 3.3 Design Pattern | 12 |
| 3.4 Design Pattern Adopted | 14 |
| 3.5 Package Diagram | 15 |
| 3.6 Sequence Diagram | 16 |
| 3.7 ER Diagram | 19 |
| 3.8 Object Diagram - Model | 20 |
| 3.9 Class Diagram - DAO | 21 |
| 3.9 Class Diagram - Controller pre-view | 22 |
| 3.10 Global Class Diagram | 23 |
| 3.11 Class Diagram - Controller final version | 24 |
| 3.12 Class Diagram - Model final version | 25 |
| 3.13 Class Diagram - Complete Package | 27 |
| 4 Implementation | 28 |

Requirements

Functional Requirements

(Verrà assegnata priorità attraverso tre classificazioni: HP(High priority), MP(Medium priority), LP(Low priority).

Login (HP): Funzione necessaria per l'accesso al sistema, senza la quale l'utente non sarà in grado di interfacciarsi con esso.

Gestione Utente e Pagine (MP): Consente all'amministratore di sistema di gestire i dati relativi ad opere, agli utenti e alle pagine, cancellandoli o modificandoli.

Visualizzazione elenco opere (HP): Funzione di browsing base, consente all'utente di visualizzare l'elenco dei titoli delle opere nel sistema.

Visualizzazione intere opere (HP): Funzione estesa a tutti gli utenti eccetto l'utente base, consente di consultare ambo immagini e testo di tutte le opere pubblicate.

Acquisizione (HP): Consente agli acquirenti di caricare scansioni dei manoscritti nel sistema.

Validazione della Scansione (MP): Consente ai revisori delle acquisizioni di decretare se una scansione è idonea e pronta per la trascrizione.

Trascrizione (HP): Una delle operazioni più importanti del sistema, accedervi deve essere semplice per gli utenti abilitati. Essa consiste nel trascrivere, appunto, tramite un editor di testo le scansioni relative ad un'opera selezionata.

Validazione della Trascrizione (MP): Consente ai revisori della trascrizione di decidere se accettare o meno la trascrizione di un manoscritto.

Pubblicazione (HP): Consente (dopo aver superato la fase di validazione della scansione) ai revisori delle acquisizioni, oppure all'amministratore di sistema, di pubblicare il manoscritto selezionato, permettendo la visualizzazione delle immagini e delle trascrizioni corrispondenti per ogni pagina.

Assumptions

Gli utenti appena registrati al sistema verranno classificati come utente base.

I collaboratori del sistema (Trascrittori, Revisori e Acquisitori) hanno accesso ai privilegi di un utente avanzato oltre che a quelli specifici del loro settore.

La funzione di upload delle immagini nel web server si potrà effettuare solamente tramite la GUI, attraverso un apposito pulsante che permetterà di scegliere il file sul proprio client.

A seguito di un'attenta analisi della specifica e di alcune discussioni, è stato deciso che:

- è permesso agli utenti (eccetto l'utente base) visualizzare anche solo le acquisizioni dei manoscritti, se prima approvate dal revisore.
- è permesso agli utenti (eccetto l'utente base) visualizzare le trascrizioni, solo se approvate dal revisore.

Non Functional Requirements

Safety: Il committente non ha richiesto vincoli di safety, pertanto il sistema viene considerato non safety-critical. La safety sarà pertanto misurata solamente in base alla sicurezza dell'account di ogni utente.

Usability: In base all'utenza che potrebbe usufruire del sistema, abbiamo deciso di propendere per un'interfaccia grafica estremamente user-friendly.

Performance: Il sistema maneggia dati (soprattutto le immagini) molto pesanti, pertanto la performance potrebbe variare a seconda delle prestazioni degli end-system e dei collegamenti con le risorse.

Operational: Andrà garantita coerenza nella sequenzialità dei dati relativi ad un'opera, nonché la corrispondenza tra opere ed indicizzazione.

Use Case

Use Case and Actors

Verranno elencati gli attori in ordine gerarchico (quelli più in basso ereditano le funzioni di quello più in alto, oltre ad aggiungere le proprie).

1. **Utente Base:** Partecipa a use case Login e visualizzaElencoOpere, che consentono, rispettivamente, di accedere al sistema e di visualizzare l'elenco delle opere pubblicate.
2. **Utente Avanzato:** Può, in addizione alle funzioni dell'utente base, visualizzare ogni opera pubblicata per intero.
3. **Acquisitore:** Può, in addizione alle funzioni dell'utente avanzato, utilizzare la funzione upload e visualizzare le opere non ancora pubblicate.

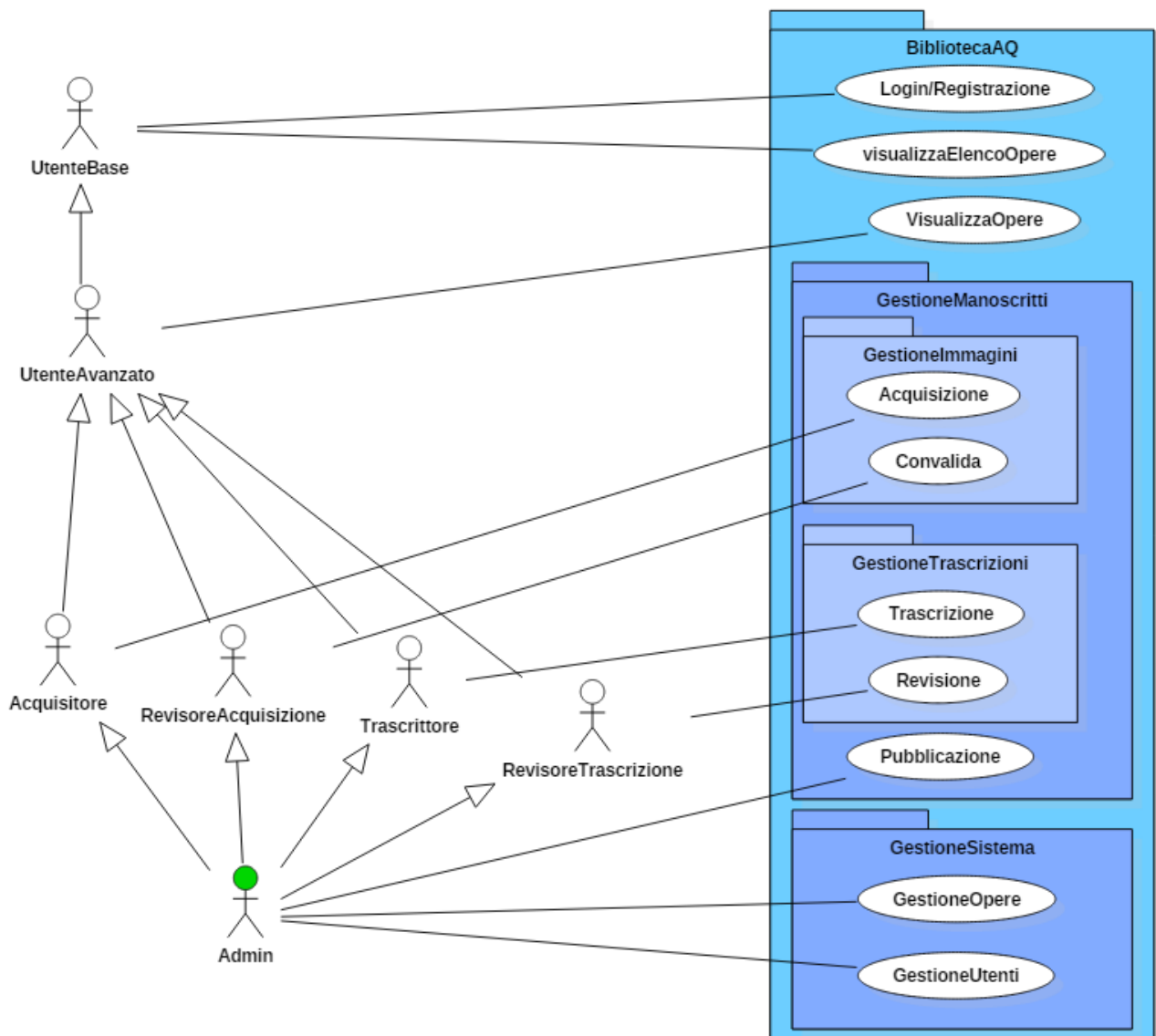
Trascrittore: Può, in addizione alle funzioni dell'utente avanzato, utilizzare la funzione trascrizione con il relativo editor di testo e visualizzare le opere non ancora pubblicate.

Revisore Acquisizioni: Può, in addizione alle funzioni dell'utente avanzato, eseguire il check sulle immagini scannerizzate (use case Convalida) e visualizzare le opere non ancora pubblicate.

Revisore Trascrizioni: Può, in addizione alle funzioni dell'utente avanzato, eseguire il check sulle trascrizioni dei manoscritti (use case Revisione) e visualizzare le opere non ancora pubblicate.

4. **Admin:** Utente più importante del sistema, può accedere alle funzioni di gestione utenza, gestione opere e gestione sistema.

Use Case diagram



Use Case description - Accesso al sistema e gestione manoscritti

| | |
|--------------------------|---|
| Use Case | Login/Registrazione |
| Attori coinvolti | Tutti gli utenti |
| Evento scatenante | Click sul pulsante di login/registrazione della GUI |
| Funzione | Permette ad un utente non registrato di procedere con il signup, mentre ad un utente già registrato, di entrare nel sistema con i propri dati |

| | |
|--------------------------|---|
| Use Case | visualizzaElencoOpere |
| Attori coinvolti | Tutti gli utenti |
| Evento scatenante | Funzione base attivata subito dopo il login |
| Funzione | Funzione base che consente di visualizzare la lista dei titoli dei testi presenti nel sistema |

| | |
|--------------------------|--|
| Use Case | VisualizzaOpere |
| Attori coinvolti | Utente Avanzato |
| Evento scatenante | Click sul titolo di una specifica opera e sull'apposito tasto Visualizza |
| Funzione | Consente di visualizzare testo e immagini presenti nel sistema relativi a quello specifico manoscritto |

| | |
|--------------------------|--|
| Use Case | Acquisizione |
| Attori coinvolti | Acquisitore |
| Evento scatenante | Upload di un file immagine nel sistema |
| Funzione | Consente agli addetti all'upload delle scansioni, di caricare una pagina nel sistema |

| | |
|--------------------------|--|
| Use Case | Convalida |
| Attori coinvolti | Revisore Acquisizioni |
| Evento scatenante | (GUI) Click sull'apposito tasto per la revisione |
| Funzione | Permette di approvare un'immagine caricata da un Acquisitore |

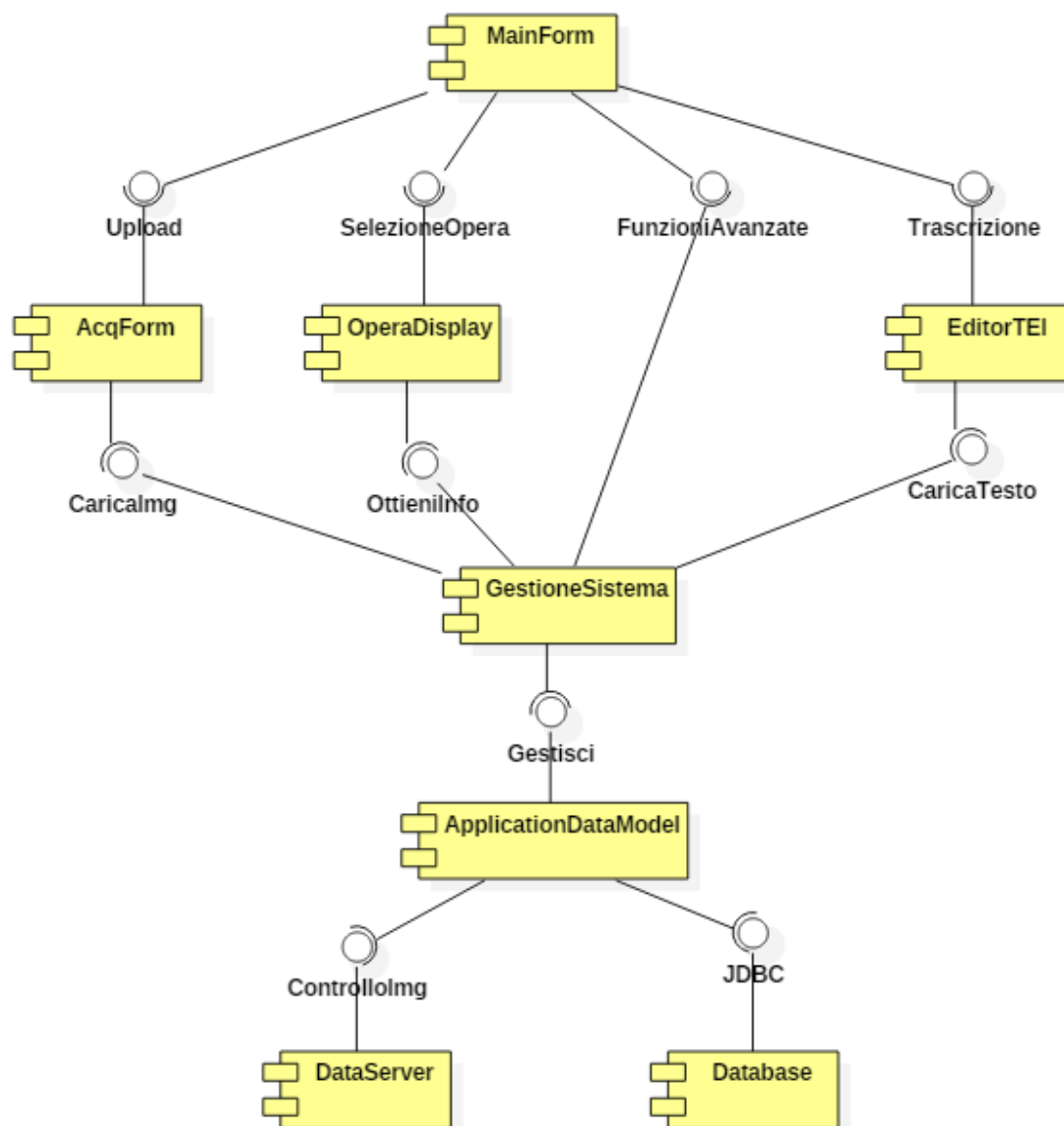
| | |
|--------------------------|---|
| Use Case | Trascrizione |
| Attori coinvolti | Trascrittore |
| Evento scatenante | Inserimento di testo nell'editor TEI aperto tramite la shortcut nella GUI |
| Funzione | Consente agli utenti idonei di accedere alle funzioni di scrittura tramite l'editor di testo scelto |

| | |
|--------------------------|--|
| Use Case | Revisione |
| Attori coinvolti | Revisore Trascrizioni |
| Evento scatenante | (GUI) Click sull'apposito tasto per la revisione |
| Funzione | Permette di approvare un testo scritto e caricato da un Trascrittore |

| | |
|--------------------------|---|
| Use Case | Pubblicazione |
| Attori coinvolti | Admin |
| Evento scatenante | (GUI) Utilizzo del tasto di pubblicazione |
| Funzione | Consente di rendere un'opera visibile anche agli utenti "base" e "avanzato" oppure lasciarla privata e non visualizzabile agli utenti al di fuori dei collaboratori al sistema. |

System Design

Component diagram - Architectural Model



Architectural choices

Abbiamo optato per la realizzazione di un'applicazione Client-Server.

La scelta è stata effettuata riflettendo sugli scenari di utilizzo del sistema, nello specifico, al tipo di utenza che l'applicazione potrebbe avere. Per questo la nostra GUI sarà molto user-friendly e veloce da utilizzare, permettendo all'utente di potersi concentrare in maniera diretta e specifica sulla sola operazione a lui permessa.

Abbiamo illustrato, nel component diagram sopra, le parti che, per questo livello di astrazione, riteniamo importanti nel sistema.

Le prime quattro componenti fanno riferimento alla "view" (spiegheremo in seguito in dettaglio a cosa facciamo riferimento) del nostro sistema. Queste sono: la "main form", la "acq form", l'editor TEI ed "opera display".

Fanno tutte riferimento ad aree della GUI, ed ognuna consente di accedere ad una delle funzioni critiche indicate prima nella descrizione degli use case.

La componente "Gestione Sistema" è quella che si identifica con i "controller" (ne discuteremo sempre più avanti parlando dei Design Pattern) e consente di interfacciarsi con il livello di "application data model", ossia il database ed il server.

1. Editor TEI

Tra le scelte nelle quali il team ha riscontrato più difficoltà nel risolvere, c'è sicuramente quella relativa all'editor di testo. Ci siamo imbattuti, dopo svariate ricerche, in JEdit. Un software interamente java-based ed in grado di riconoscere e validare TEI attraverso l'utilizzo dei Plug-In XML, ErrorList ed XLST per trasformare documenti XML usando le stylesheets.

Per schemi diversi da RNG va aggiunto un file schemas.xml nella directory locale oppure installarlo globalmente nella cartella dei plugin XML aggiungendo queste 4 righe di codice (scelta consigliata):

```
<?xml version="1.0" ?>

<locatingRules xmlns="http://thaiopensource.com/ns/locating-rules/1.0">

  <namespace ns="http://www.tei-c.org/ns/1.0" uri="tei_all.rng"/>

</locatingRules>
```

Attraverso il plugin XML si possono anche generare DTD per la validazione TEI.

2. TEI Viewing

Per visualizzare correttamente il TEI in un JFrame abbiamo dovuto affrontare non poche problematiche.

Il primo problema riguardava la memorizzazione del file nel DB poiché come scelta implementativa abbiamo deciso di utilizzare il web storage esclusivamente per le immagini e non per i file XML. Per evitare errori nelle query di inserimento e lettura del file a causa dei tag, abbiamo optato per codificare in base 64 il contenuto della trascrizione:

Scrittura) File XML -> FileReader -> Encode base64 -> PreparedStatement

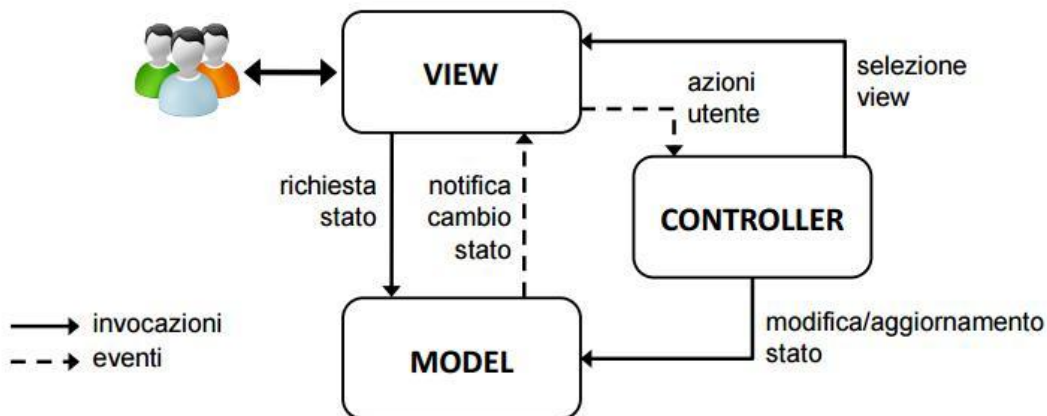
Lettura) Statement -> Decode base64

Il secondo problema, sicuramente il più complesso, riguardava il convertire il TEI in un formato ideale per la sua corretta visualizzazione. Siccome il plain text a nostro parere non era adatto per un'applicazione in cui la visualizzazione delle trascrizioni è fondamentale, abbiamo scelto l'HTML. Per renderlo possibile abbiamo utilizzato le classi Transformer base con l'aggiunta delle librerie Saxon in combinazione con gli StyleSheet XSLT per l'HTML.

L'ultimo problema, ma non meno importante, era la visualizzazione dell'HTML generato dal file TEI. JFrame supporta l'html base ma a nostro parere non era sufficiente per una visualizzazione completa. Per questo motivo abbiamo utilizzato il JFXframe richiamato in un thread separato, adatto a questo scopo, per l'html5.

Design Pattern - MVC (Model View Controller)

MVC è un pattern architetturale che descrive il più alto livello di astrazione di un sistema software. Questo, nello specifico, consente di separare e disaccoppiare il modello dei dati (model) e la logica applicativa (controller) dalle modalità di visualizzazione e interazione con l'utente (view).



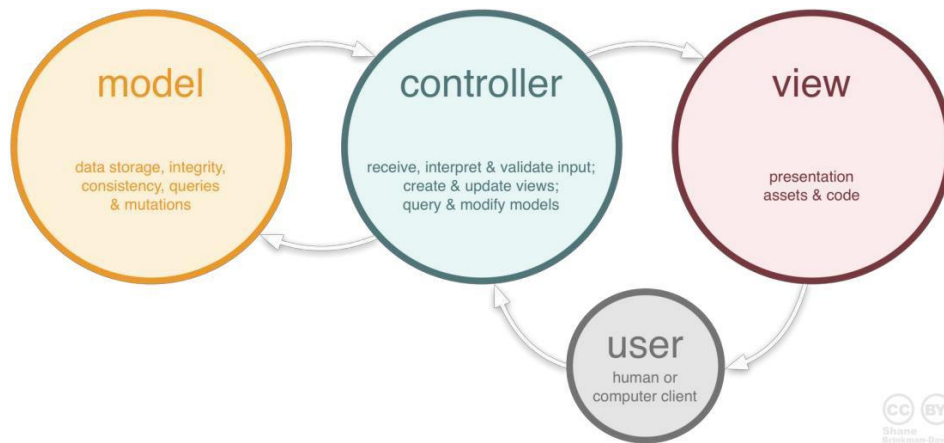
In Java si ha che:

- l'interazione tra view e controller avviene in base al meccanismo di propagazione e gestione eventi Swing/AWT.
- i componenti controller sono `EventListener` (es. `ActionListener`, `MouseListener`...) associati ai componenti grafici della View (es. `JButton`).

Le componenti del nostro sistema andranno ad identificarsi in tre ruoli principali: Model, View e Controller.

- L'utente interagisce con la View → Il Controller riceve la sua azione e la interpreta.
- Il Controller comunica al Model di cambiare il suo stato.
- Il Controller fa cambiare stato anche alla View.
- Il Model segnala alla View che il suo stato è cambiato.
- La View richiede e riceve le informazioni sul nuovo stato del Model.

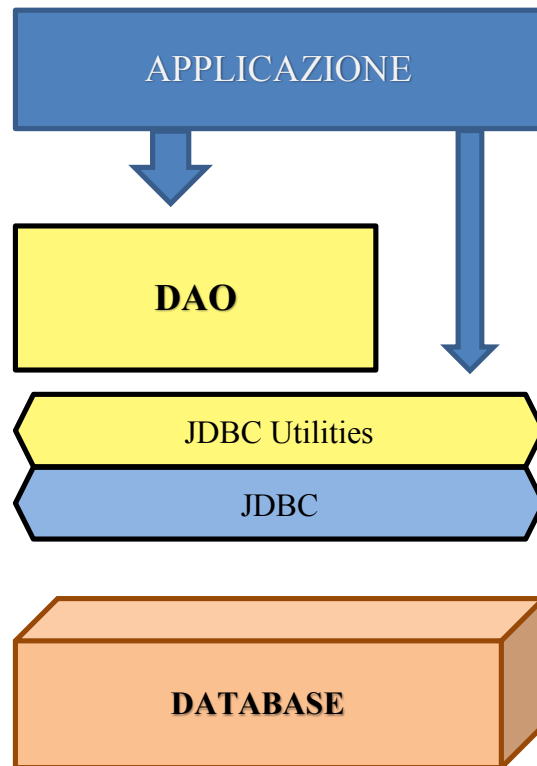
1. Why MVC



MVC perchè volevamo un sistema loose coupled e highly cohesive, in modo che in caso di future modifiche al software, il programmatore sarebbe notevolmente facilitato, risultando il codice più comprensibile e le componenti principali facili da individuare.

Fondamentalmente, le proprietà che ci fornisce MVC, sono: estendibilità, riusabilità, interoperabilità. Inoltre si possono anche notare delle qualità interne: strutturazione, modularità (come già specificato in precedenza), comprensibilità e manutenibilità.

Design Pattern Adopted - DAO

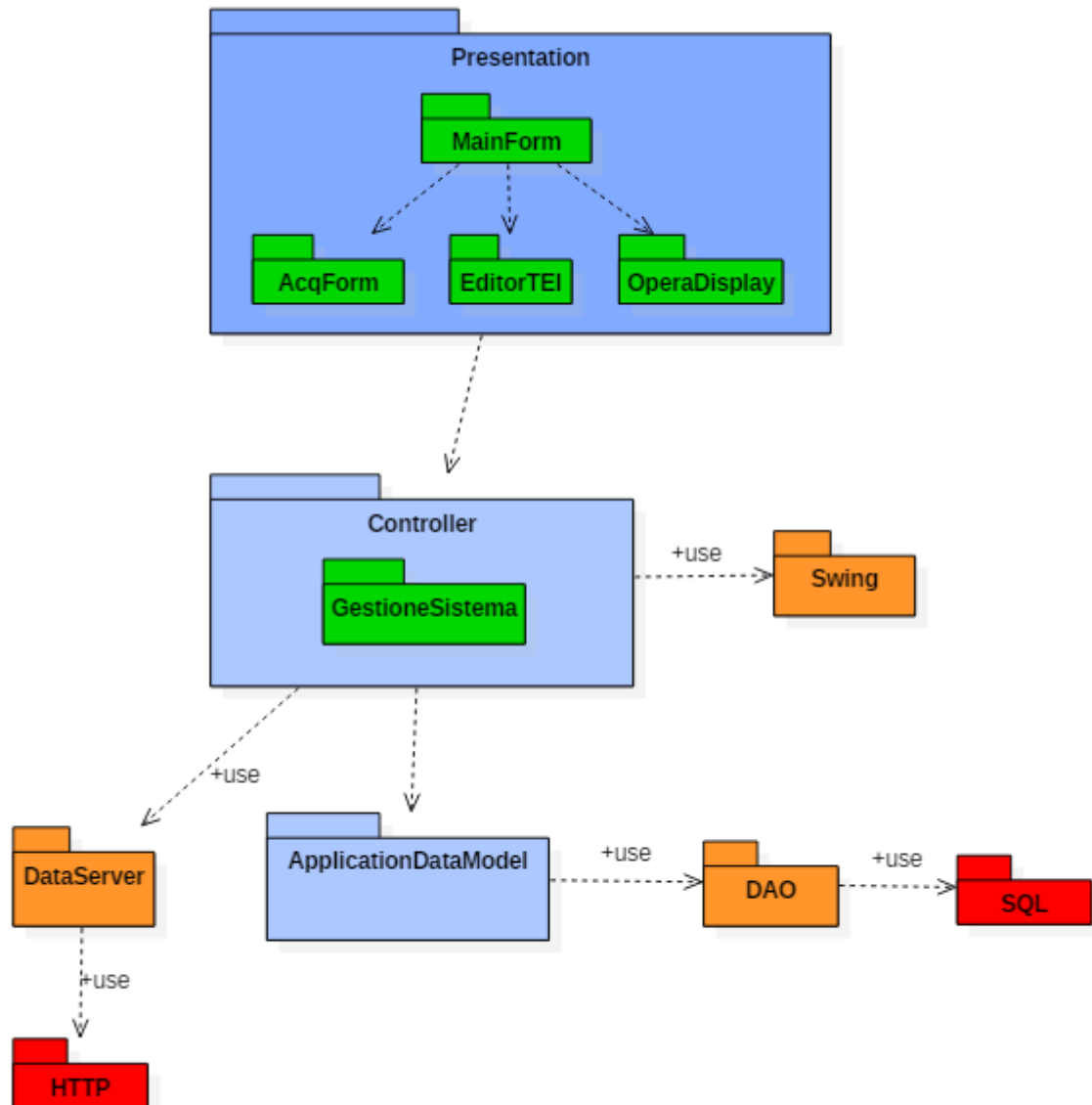


Abbiamo sfruttato questo pattern di cui si è discusso nel corso, in quanto ci permette, isolando l'accesso al "data layer" da parte della "business logic" (ossia isolare l'accesso ad una tabella tramite query poste all'interno dei metodi della classe), di avere facilitazioni nella manutenzione ed un maggiore livello di astrazione.

In questo modo abbiamo che solo gli elementi del DAO possono interagire con database e server, ottenendo un sistema più organizzato e preciso.

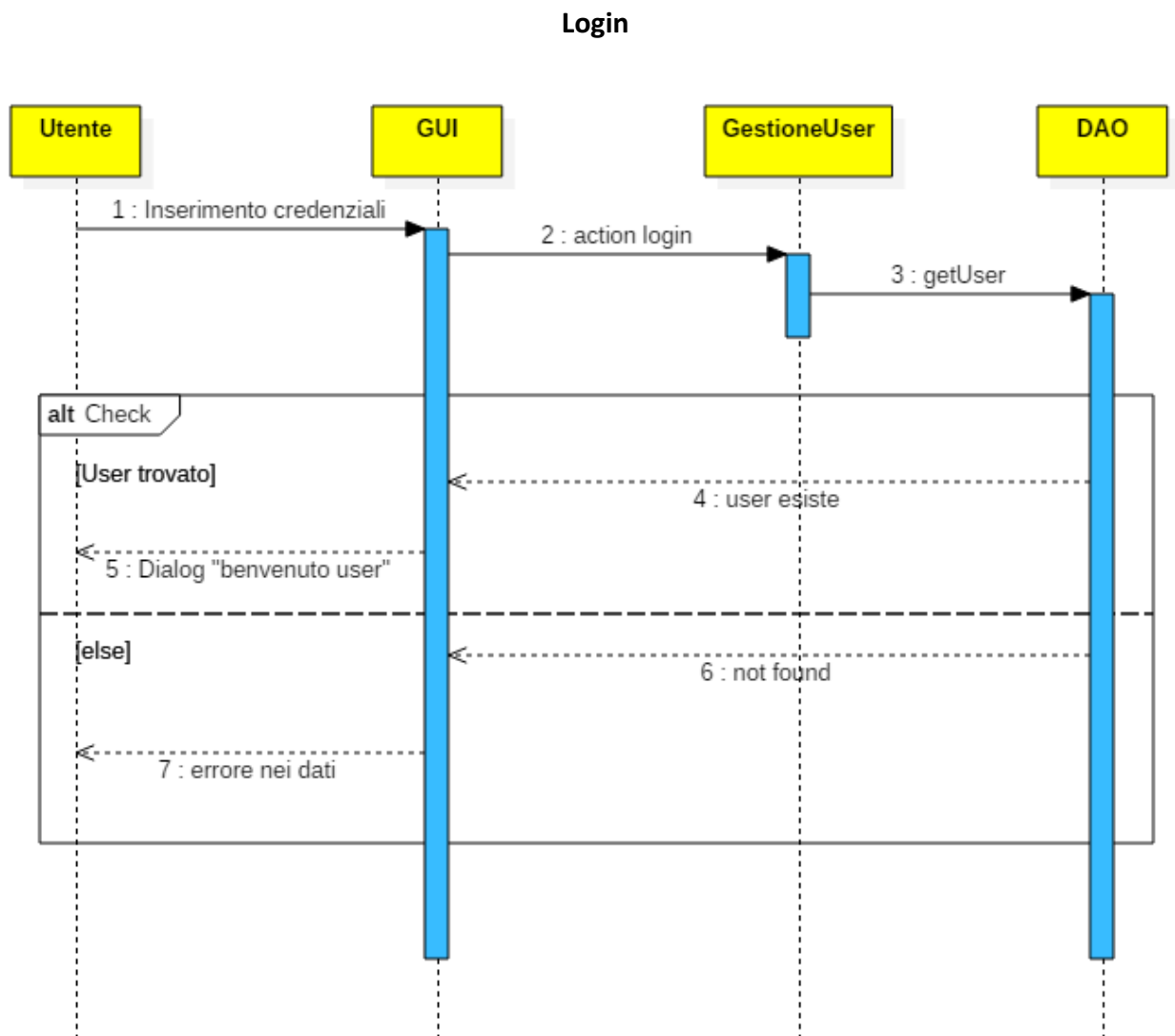
La scelta è sempre correlata a quella fatta per MVC, volendo favorire scalabilità e manutenibilità del software.

Package Diagram

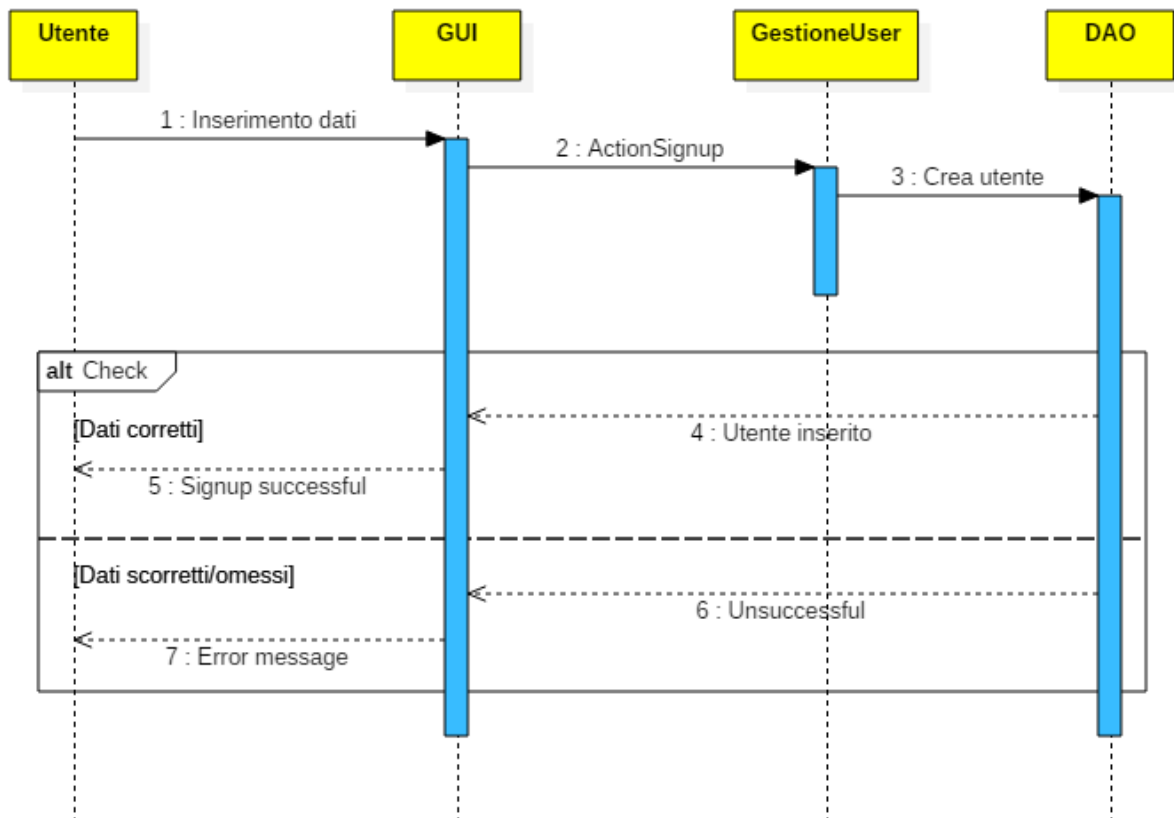


Sequence Diagrams

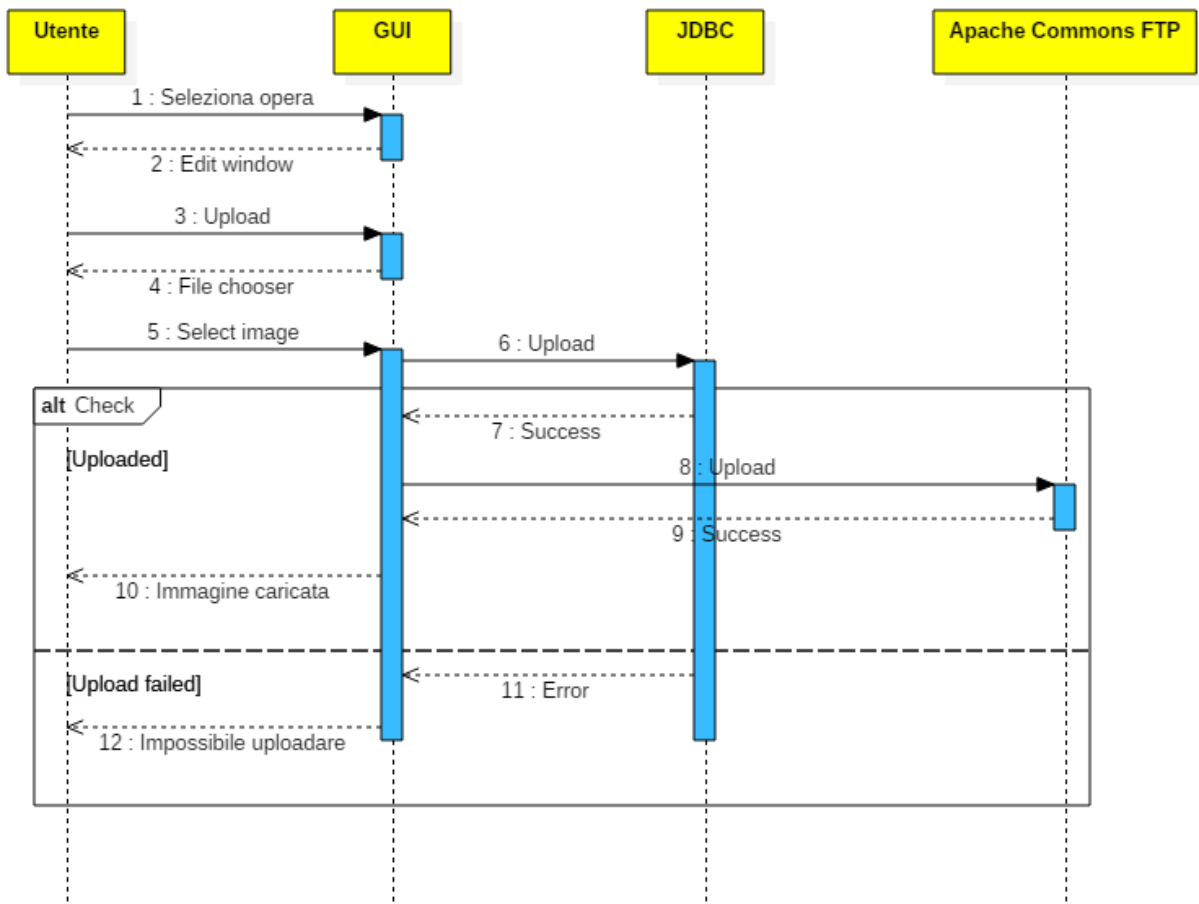
Di seguito verranno mostrati alcuni scenari del nostro sistema, attraverso dei sequence diagram.



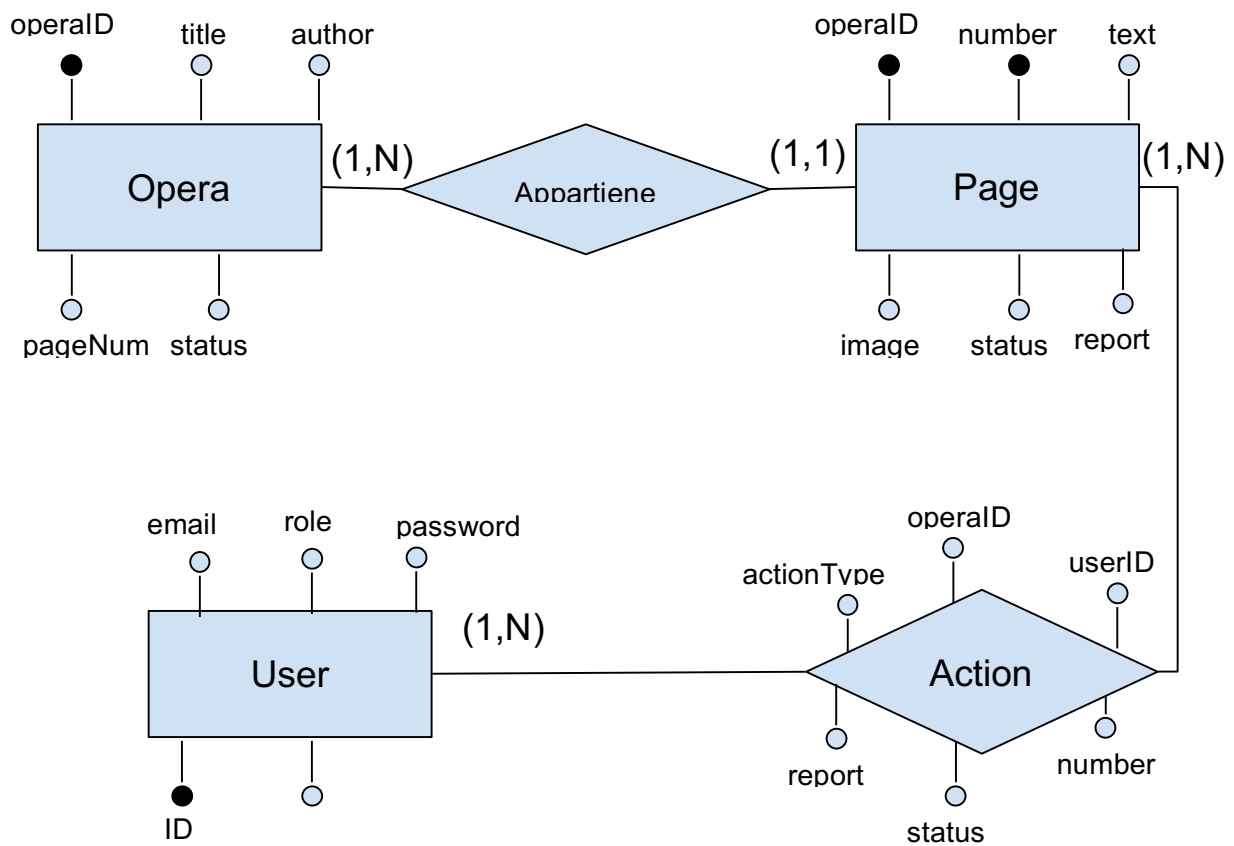
Registrazione



Caricamento immagine



ER Diagram



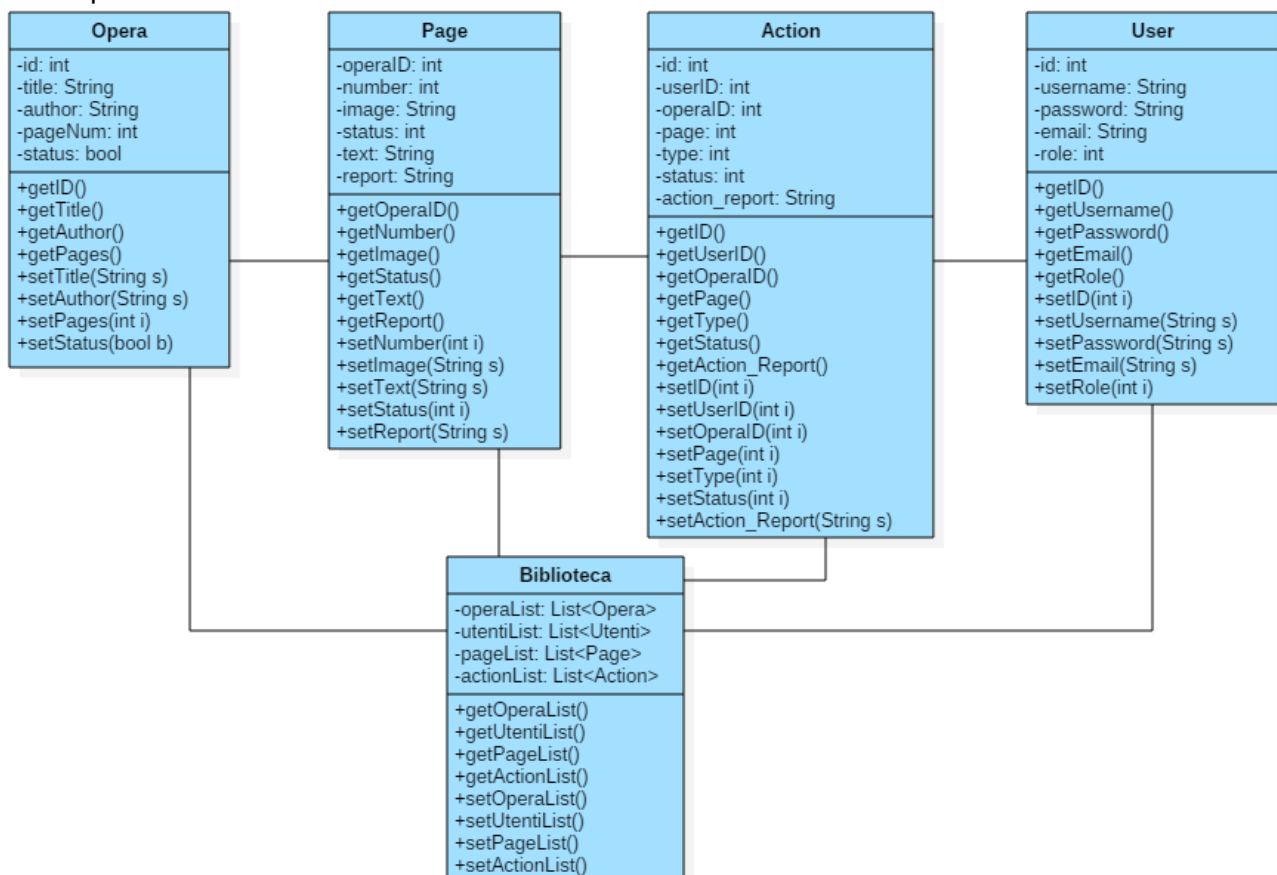
Object Diagram - Model

L'oggetto Opera è il manoscritto nella sua interezza. L'id la rappresenta univocamente, è composto poi dai dati relativi al manoscritto (titolo, autore e numero di pagine), mentre lo status è un booleano che indica l'avvenuta pubblicazione o meno.

L'oggetto Page indica la singola pagina del manoscritto. Fa riferimento all'id dell'opera a cui appartiene ed il numero è la sua collocazione all'interno di essa. Ogni Page è composta da un'immagine e l'eventuale testo trascritto. Lo status indica quali di queste due componenti è presente e convalidata.

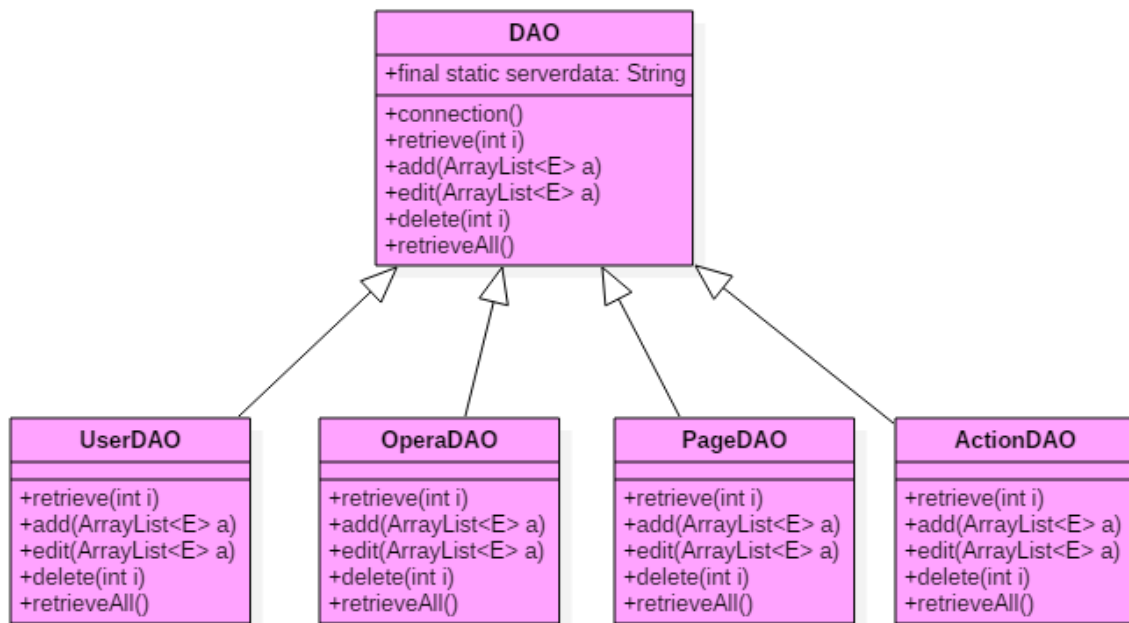
L'oggetto Utenti identifica coloro che sono registrati al sistema. Sono univocamente rappresentati da un id, mentre i restanti dati sono relativi al login (username, password ed email) ed alla loro funzione nel sistema (role).

L'oggetto Action identifica le operazioni che l'utente esegue su ogni pagina di un'opera.
L'oggetto Biblioteca ci permette di accedere alla totalità delle opere, degli utenti, delle pagine e delle operazioni nel sistema.



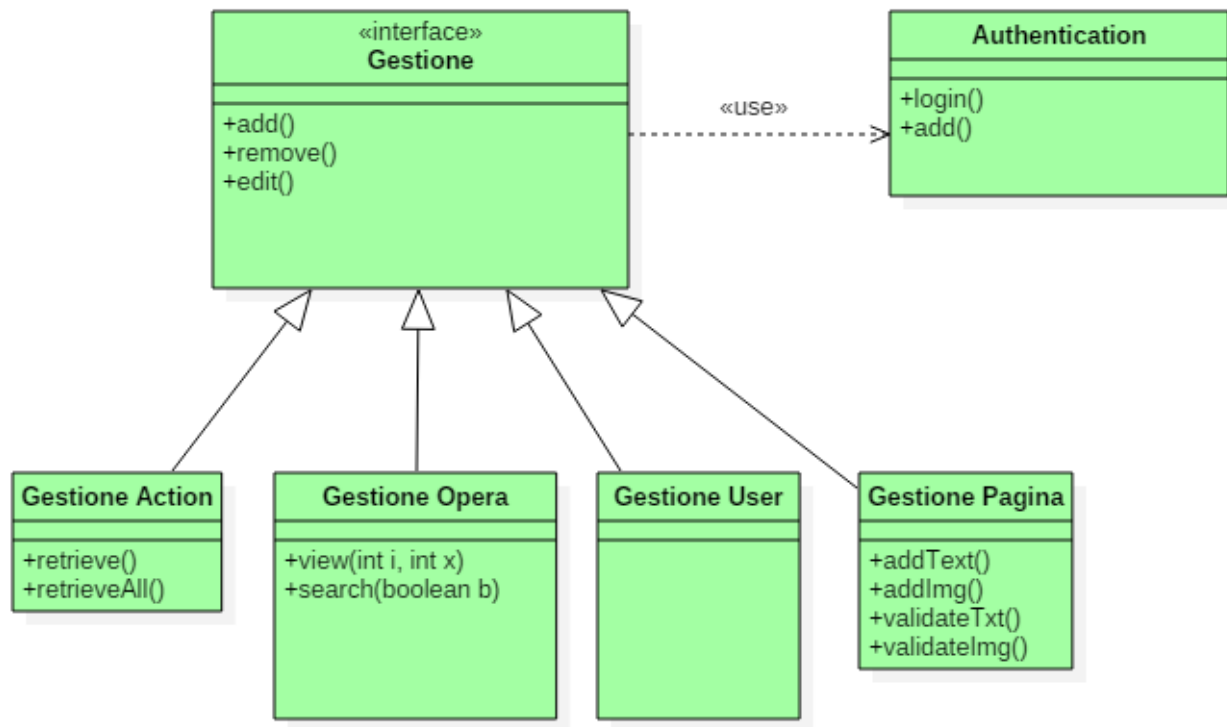
Class Diagram - DAO

Il seguente class diagram mostra come le nostre classi DAO interagiranno col DB.



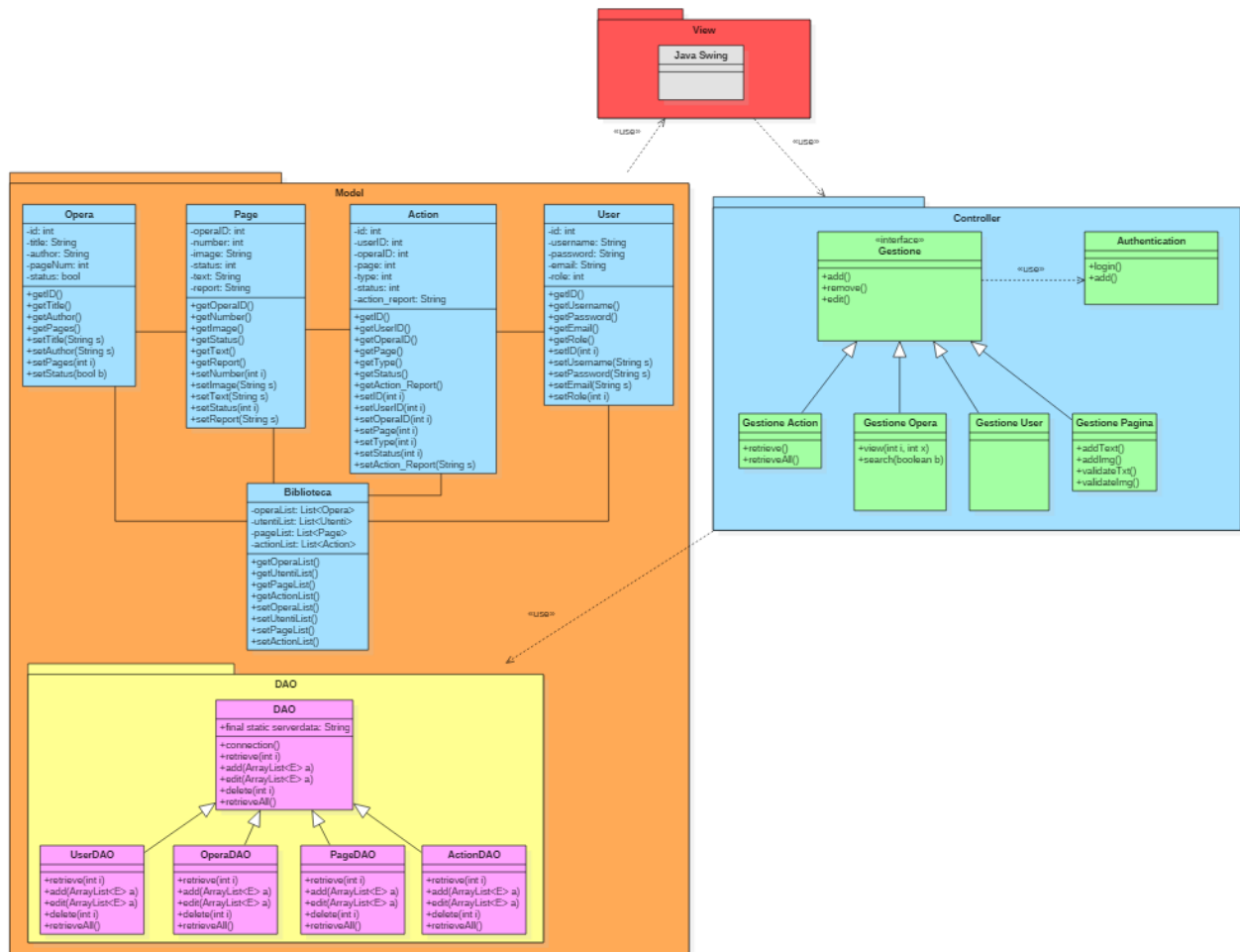
Class Diagram - Controller pre-view

Anteprima di alcune funzioni della classe Controller che, è la più complessa sotto tutti gli aspetti.

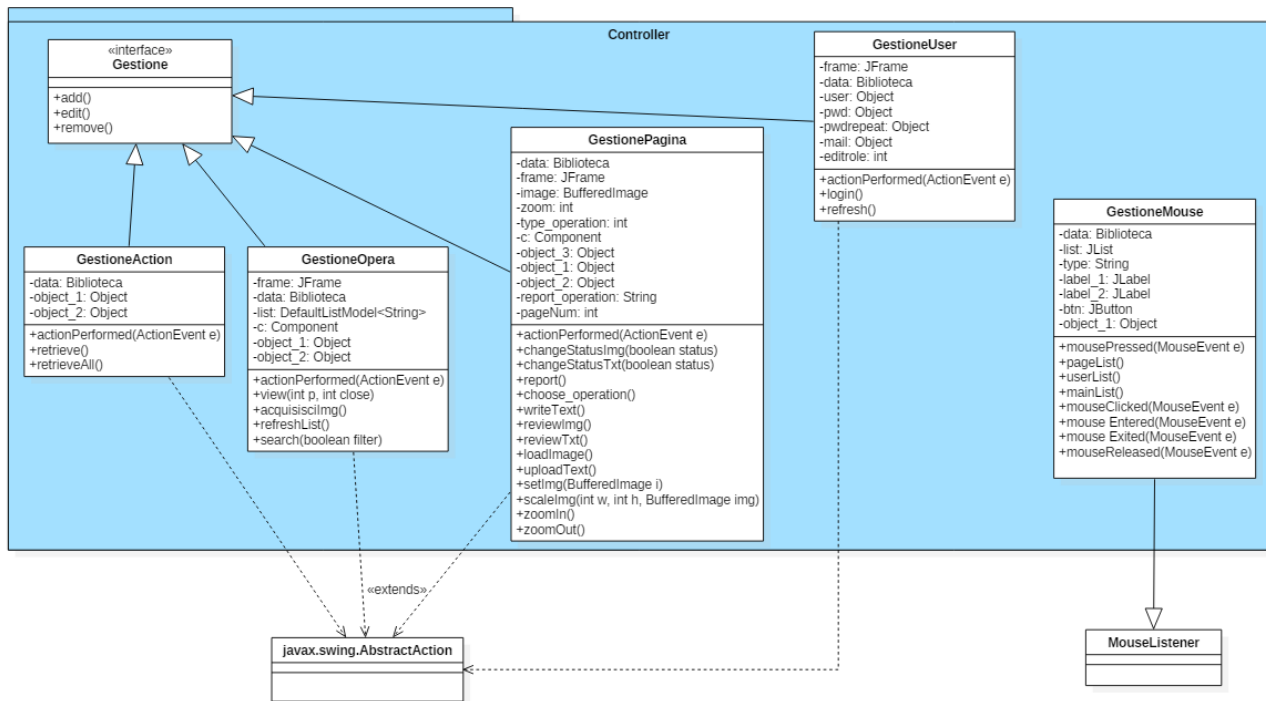


Global Class Diagram

Il seguente Class diagram rappresenta la visione generale del sistema, mostrando come gli oggetti interagiscono tra di loro, a partire da Java Swing.



Class Diagram - Controller final version

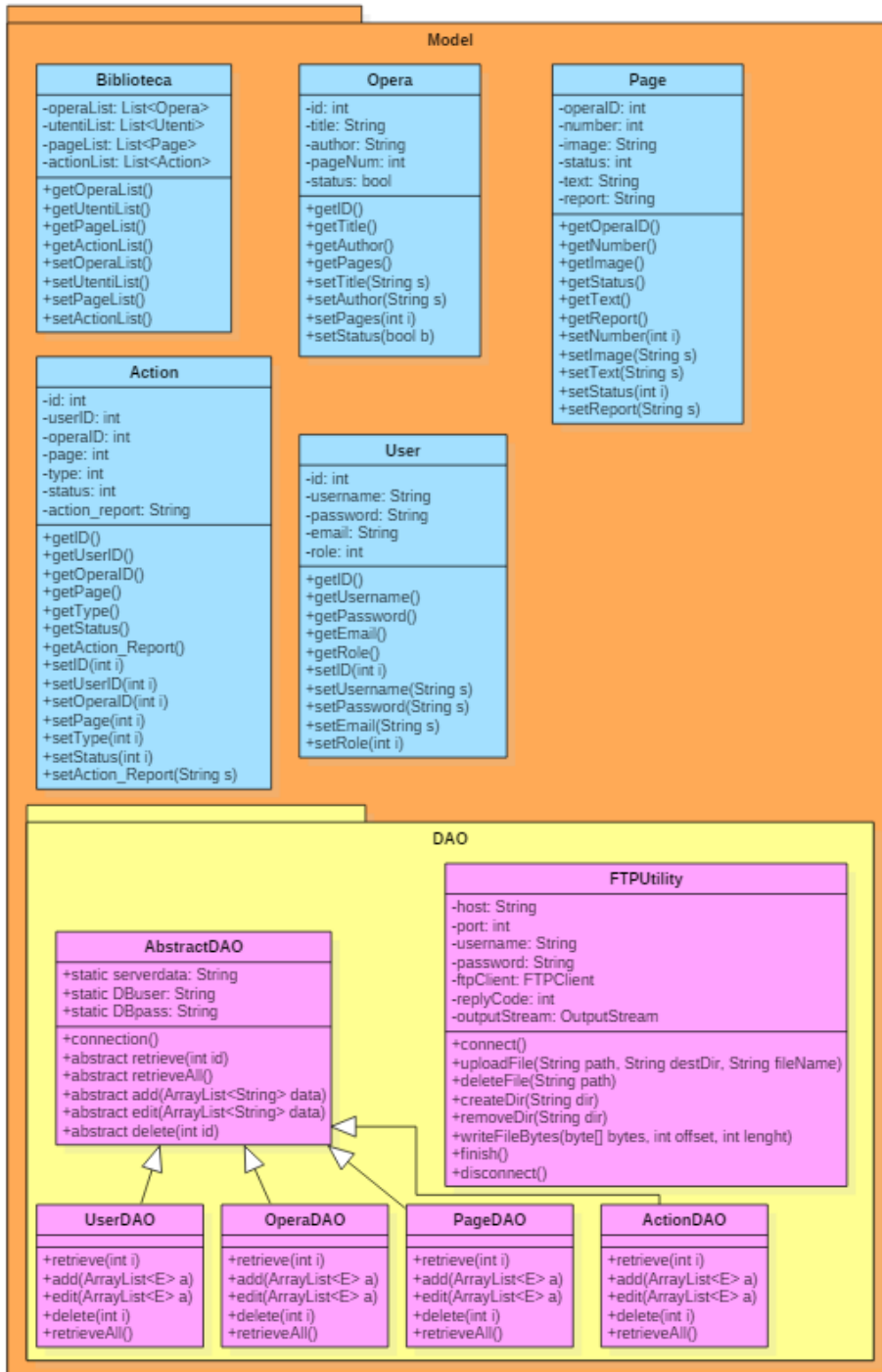


Versione definitiva del Package Controller, in cui catturiamo tutte le azioni compiute, dall'utente o dal sistema stesso, nella view.

Swing di norma prevede che la gestione delle action vengano fatte direttamente nelle view ma, noi non possiamo permettere questo per via della scelta pattern MVC. Dunque, per evitare che andassimo fuori pattern, abbiamo definito delle classi nel package Controller per estendere `AbstractAction` così da trattarle come action listeners. Questo è anche il principale motivo per cui molte delle classi prendono come argomenti degli oggetti generici, essendo richiamate da action completamente differenti fra loro anche se applicate allo stesso model.

Tutte le classi implementano l'interfaccia **Gestione** e le relative funzioni, fatta eccezione per **GestioneMouse** che implementa **MouseListener** e viene utilizzata esclusivamente per mostrare alcuni dettagli dopo l'interazione con le liste.

Class Diagram - Model final version



Come specificato precedentemente, in questo package vengono dichiarate tutte le classi utili al Controller per la gestione dati presenti sul DB.

Le classi DAO si interfacciano con il DB, manipolando i dati in esso presenti in base alle richieste degli utenti.

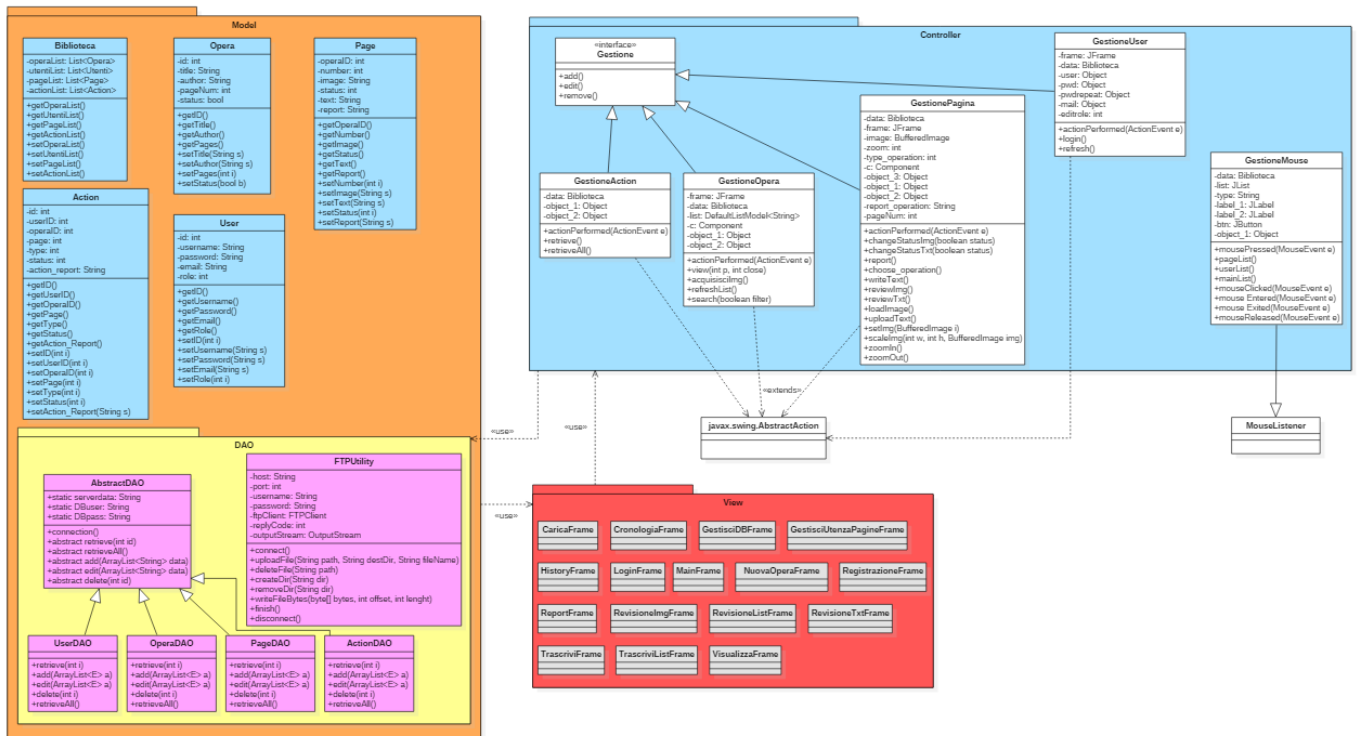
AbstractDAO è la classe che si occupa della connessione al DB, ottenendo le credenziali dal file di configurazione locale (semplice file di testo) presente nella stessa source path del programma.

Il suddetto file viene utilizzato solo alla prima connessione al database durante il lancio dell'applicazione, scelta dovuta al fatto che l'Admin del sistema può cambiare, in qualsiasi momento, DB mediante il pannello di gestione database.

Si distingue dalle altre la classe FTPUtility che utilizza la libreria Apache Commons per effettuare la manipolazione dei dati sul server online dove sono contenute le acquisizioni delle opere.

Contiene funzioni utili al popolamento, modifica e cancellazione dei singoli file o di intere cartelle.

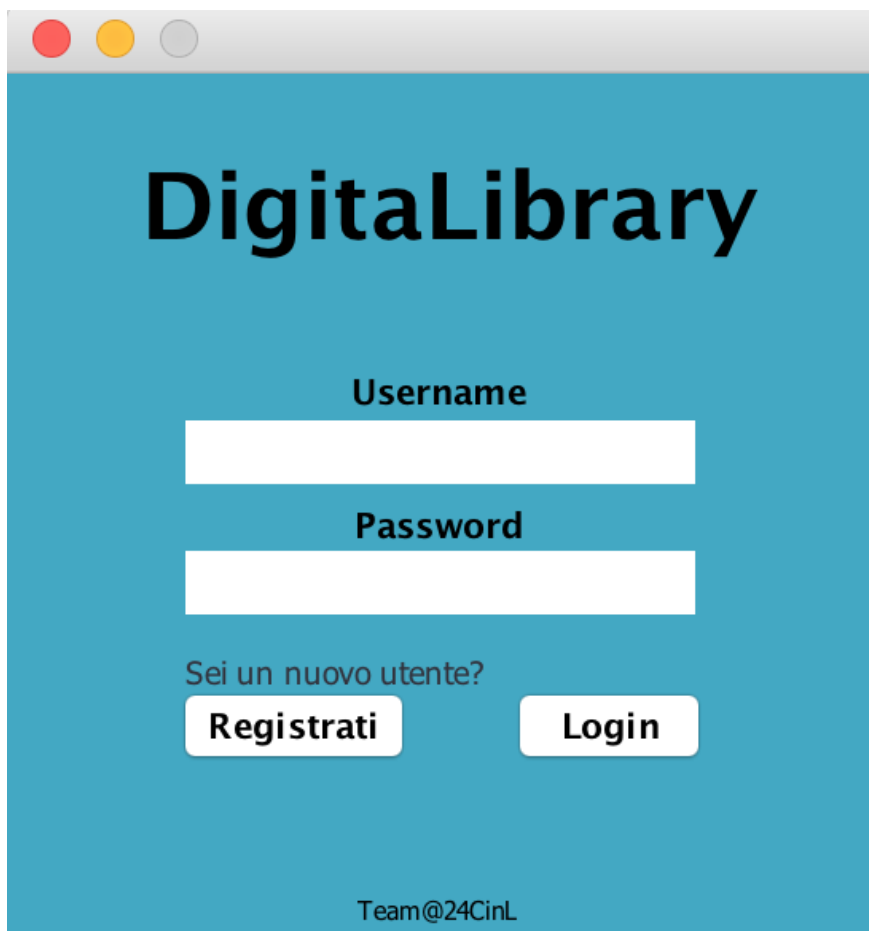
Class Diagram - Complete Package



Implementation

Abbiamo deciso di mostrare solo alcuni screen riguardanti la nostra applicazione durante l'esecuzione.

Login

A screenshot of a web application window titled "DigitaLibrary". The window has a blue background and a white title bar with three colored buttons (red, yellow, grey). The main content area is blue and contains the following elements: the title "DigitaLibrary" in large black font; a "Username" label above a white input field; a "Password" label above a white input field; a link "Sei un nuovo utente?" in grey text; two white buttons labeled "Registrati" and "Login"; and the text "Team@24CinL" at the bottom center.

DigitaLibrary

Username

Password

Sei un nuovo utente?

Registrati **Login**

Team@24CinL

Amministratore - Main window

DigitaLibrary

Cerca

☒ Titolo ☐ Autore

Elenco opere

- Divina Commedia
- Iliade
- Odissea
- Decameron

Anteprima opera

Status: -
Autore: -
Pagine: -

Visualizza

Operazioni Privilegiate

Pannello operazioni privilegiate, permesse esclusivamente all'amministratore di sistema.

Gestione opera

- Pubblica/Privata**
- Nuova Opera**
- Elimina Opera**
- Gestisci Pagine**

Gestione Sistema/Utenza

- Gestisci Utenti**
- Gestione Database**
- Storico Operazioni**


Copyright - Team@24CinL

Opera view

Pagina 1

Divina Commedia

DigitalLibrary - Team@24CinL



Pagina 1

(unknown project)

Divina Commedia

AUTORE: Dante Alighieri

TEI validate Ok.

[Home](#)

Date: 2016-11-29

Vai a

Go

Pagina 1